Performance Evaluation of LED-to-Camera Communications

Alexis Duque, Adrien Desportes, Razvan Stanica, Hervé Rivano

Performance Evaluation of LED-to-Camera Communications

Alexis Duque
Adrien Desportes
Rtone
Lyon, France

Razvan Stanica
Hervé Rivano
Univ Lyon, INSA Lyon, Inria, CITI
Villeurbanne, France

ABSTRACT

The use of LED-to-camera communication opens the door to a wide range of use cases and applications, with diverse requirements in terms of quality of service. However, while analytical models and simulation tools exist for all the major radio communication technologies, the only way of currently evaluating the performance of a network mechanism over LED-to-camera is to implement and test it. Our work aims to fill this gap by proposing a Markov-modulated Bernoulli process to model the wireless channel in LED-to-camera communications, which is shown to closely match experimental results. Based on this model, we develop and validate *CamComSim*, the first network simulator for LED-to-camera communications.

KEYWORDS

visible light communications, optical camera communications, Markov-modulated Bernoulli process, simulation

1 INTRODUCTION

Visible-light communication (VLC) is an enabling technology that exploits illumination to provide a short-range wireless communication link. VLC systems take advantage of the license-free light spectrum and their immunity to radio frequency (RF) interference. In such systems, information is often relayed by modulating the output intensity of a light-emitting diode (LED). Any electronic device which can detect the presence or absence of visible light can be utilized as a VLC receiver. While most of the work in the field is focused towards using photo-diodes as receivers, because of their fast response and high bandwidth, some studies demonstrated that smartphone cameras can also be used to detect high-frequency light patterns [1].

Indeed, nowadays smartphone cameras widely use two types of image sensors, Charge Coupled Devices (CCD) or Complementary Metal Oxide Semiconductors (CMOS). These two technologies have some similarities, but one major distinction is the way each sensor exposes its pixels to light. CCD sensors use the Global Shutter readout mode, where all pixels are exposed simultaneously and then each pixel is read sequentially. This mechanism helps in capturing a still image of a moving object. On the other hand, CMOS sensors use the Rolling Shutter readout mode [2], where each row is exposed in a row-sequential way with fixed time delay. Due to this mechanism, there is a significant time difference between the beginning of the exposure of the first and the last row, making them no longer simultaneous. When an LED is modulated at a frequency higher than the rolling shutter speed, stripes of different light intensity are captured in the image. A row of pixels appears illuminated when the LED was ON during the row exposure time. On the other hand, a row appears dark when the LED was OFF during the exposure time. The intensity and width of the strip depend on the transmitter modulation frequency, allowing us to encode information in these illuminated and dark bands, similarly to the use of a bar code.

This LED-to-camera communication based on the Rolling Shutter effect opens the door to a wide range of use cases and applications, with diverse requirements in terms of quality of service [3]. To cite a few examples, both line-of-sight (LOS) [4, 5] and non-line-of-sight (NLOS) [6, 7] communications have been demonstrated in these settings, as well as ultra-reliable localization solutions [8, 9], sensing [10], or even scene protection against intrusive photographs [11]. However, while analytical models and simulation tools exist for all the major RF technologies, the only way of currently evaluating the performance of a network mechanism over LED-to-camera is to implement and test it. This results in heavy measurement and parameterisation campaigns that need to be repeated anytime a new VLC protocol or feature is imagined. Having access to standard performance evaluation tools in this type of network would certainly accelerate studies in the field, and nicely complement experimental field tests.

The work described in this paper aims to fill this gap by proposing models and tools that help in the assessment of LED-to-camera communication network mechanisms. Our contributions are threefold. First, we propose an analytical model for LED-to-camera communication systems in Sec. 3. Second, we design and implement *CamComSim*, a LED-to-camera communication simulator in Sec. 5. Finally, the model and the simulator allow us to benchmark several network redundancy mechanisms proposed in the literature. By checking against experimental results, we are able to confirm the correctness of our models in this context.

2 RELATED WORKS

LED-to-camera communication allows for low-throughput unidirectional message transmission, with a performance in range of a few kbit/s [4, 5]. Despite this limited throughput, LED-to-camera communication raises a lot of interest because it enables communication with no extra financial cost between any LED-equipped machine or instrument and any regular smartphone. This low cost also explains the fact that practically every study on this topic uses an experimental approach. While experiments are essential in evaluating new protocols and services, running an experimental campaign every time one wishes to evaluate a new idea can become cumbersome.

Designing analytical models and implementing them in simulation tools is standard practice in the wireless networking field in order to accelerate the evaluation of new protocols and mechanisms. Indeed, network simulation has been largely studied in the case of wireless communication [12]. Nonetheless, VLC performance evaluation remains poorly investigated and VLC simulation tools are still missing. The main efforts on simulating VLC systems have focused on indoor channel simulation [13], or on the 802.15.7 PHY [14, 15]

and MAC [16] layers. These approaches rely on classical network simulation frameworks, such as those used for wireless and ad hoc networks, e.g. ns-2 [16], ns-3 [14], OMNET++ [15] or MATLAB [13]. All these works consider LED-to-Photodiode communication, hence LED-to-Camera communication is completely unexplored. Our work is the first effort in LED-to-Camera simulation reported in the literature, making CamComSim the first implementation of a LED-to-Camera VLC simulator.

3 MODELING LED-TO-CAMERA COMMUNICATION

In this section, we describe, for the first time in the literature from our knowledge, a LED-to-camera communication channel model. Based on the theory of Markov-modulated Bernoulli processes (MMBP) [17], discussed in Sec. 3.1, this model can be applied to all LED-to-camera communication systems, not only to the particular case of ours. The proposed model is not only generic, but also very accurate, as demonstrated by its validation with extensive experimental results in Sec. 3.4. As an example, we use the analytical model to compare two simple redundancy mechanisms, required to cope with the inherent losses of LED-to-camera communication and described in Sec. 3.2 and Sec. 3.3.

3.1 Model design

In a LED-to-camera system, data is received as a series of dark and illuminated stripes in a picture frame captured by the camera. In the following, we note by f_i the i-th frame captured by the camera and by δ_f the time between the beginning of two consecutive frames. Obviously, even at the highest frame rate allowed by the camera, data is not continuously received, as a minimum time δ_g exists between two frames. This is denoted as the inter-frame gap (IFG). Moreover, as depicted in Fig. 1, the distance between the LED and the camera also has an impact: when the camera is farther away, the LED transmission is captured for a shorter time, resulting in a smaller ROI.

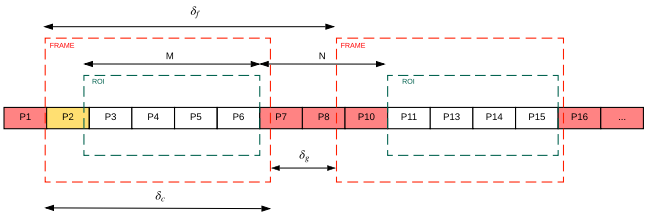

Figure 1: Frame capture time and inter-frame interval, and their relation with the MMBP parameters.

3.1.1 Gilbert-Elliot model: A first idea to model LED-to-camera communication would be the Gilbert-Elliot model, which is widely used to model bursty losses [18]. This unique type of channel can intuitively be modeled by a two states Markov chain. In state S_1, the system is capturing a frame. The camera is receiving packets, and the reception probability is $1 - p_e$ where p_e is the packet decoding error probability. In state S_2, the camera is not capturing any pictures, therefore we consider the reception probability is 0. The transition probability between S_1 and S_2 (respectively S_2 and S_1) is denoted p (respectively q). The model assumes that p, q, p_e are independent and constant.

In this case, the probability of being in state S_1 under the steady-state regime can be easily computed as $p_{s_1} = \frac{p}{p+q}$. The probability of being in state S_2 is so $p_{s_2} = \frac{q}{p+q}$.

The values of p and q are function of the duration of the IFG, δ_g, the frame duration, δ_f, and the camera capture time δ_c, depicted in Fig. 1, and linked as the following:

$$p = 1 - q = \frac{\delta_f - \delta_g}{\delta_f} = \frac{\delta_c}{\delta_f} \tag{1}$$

3.1.2 MMBP model: If the model introduced just before is straightforward and widely used, it lacks realism in our case, where the transitions between ON and OFF states are almost deterministic. Practically, in our system, the transition probability from a state to another depends on the residence time in this state.

To improve this approach, we model the LED-to-camera channel using a Markov-modulated Bernoulli process (MMBP), represented in Fig. 2. In this figure, we depict a Markov chain with a total number of $M+N$ states. Each of these states represents a reception time slot, i.e. the time duration needed in order to receive one physical layer message (denoted as PHY-SDU in the following). The transition between two states representing successive time slots is automatic, i.e. it happens with a probability of 1.

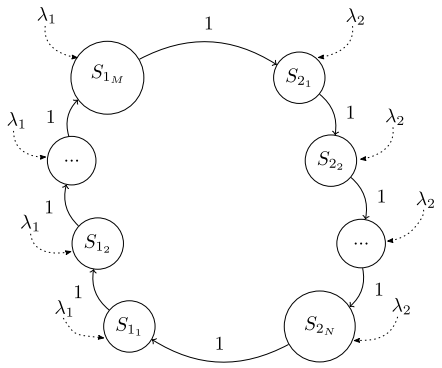

Figure 2: The MMBP model of the LED-to-camera channel.

Practically, the $M+N$ states in Fig. 2 represent a δ_f time interval, and they are divided in two groups: M states corresponding to the camera capture time δ_c (S_{ON} states), and N states corresponding to the inter-frame time δ_g (S_{OFF} states). A Bernoulli arrival process is associated with each of these $M + N$ states, representing the reception of a packet.

In S_{ON} states, the camera is receiving packets, and the arrival rate is $\lambda_1 = (1 - p_e)$, where p_e is the packet decoding error probability. In S_{OFF} states, the camera is not capturing any pictures, therefore we consider the arrival rate $\lambda_2 = 0$.

We denote as s a state in the Markov chain and we define state $s + j$ as the state reached after j transitions, starting from state s. The probability of being in state s under the steady-state regime can be easily computed as $\pi_s = \frac{1}{N+M}$. At the same time, the probability

of noticing no arrivals (i.e. no packet reception) in state s is $p_0(s)$. This can be written as:

$$p_0(s) = \begin{cases} 1, & if \quad s \in N \\ p_e, & if \quad s \in M \end{cases} \tag{2}$$

As it can be seen from both models, the relatively high packet loss probability (compared with RF technologies) is an intrinsic property of the LED-to-camera communication channel. To overcome this problem, redundancy mechanisms are needed.

In the following, using classical redundancy mechanisms as an example, we show that the classical Gilbert-Elliot model is inaccurate, which highlights the need to rely on the MMBP theory. Then, we use the MMBP channel model to compare two simple, but widely used redundancy solutions: repeating a packet or repeating a sequence of packets.

3.2 Repeat Packet

The first strategy to cope with the inherent losses in the LED-to-camera communication system, used for example by *Ferrandiz-Lahuerta et al.* [7], is to send each packet twice in a row, to increase the probability that at least one of the transmissions will be fully captured by the smartphone camera. We generalize this approach in the Repeat Packet (RP) strategy, where each packet is repeated r times, one after the other. In this case, the r value needs to be chosen in order to attain a desired reception probability, its optimal value depending on the inter-frame time and on the packet size.

In the following, we study the probability of receiving a packet at least once when considering the RP strategy, for the two models introduced above.

3.2.1 Gilbert-Elliott Model: If we consider the Gilbert-Elliott model, the probability of receiving a packet at least once can be written as $p_s^{RP} = 1 - p_0^{RP}$, where p_0^{RP} represents the probability of failing to receive a packet r times in a row, written as:

$$p_0^{RP} = \left(P_{S_1} \cdot p_e + P_{S_2}\right)^r = \left(\frac{p \cdot p_e + q}{q + p}\right)^r \tag{3}$$

3.2.2 MMBP Model: If we consider the MMBP model, the probability of failing to receive a packet r times in a row p_0^{RP} can be written as:

$$p_0^{RP} = \frac{\sum_s \left(p_0(s) \cdot p_0(s+1) \cdot p_0(s+2) \cdot \ldots \cdot p_0(s+r-1)\right)}{N+M} \tag{4}$$

The value of p_0^{RP} will depend on m_s, defined as the number of S_{ON} states during the r retransmissions, when the first packet transmission is in state s:

$$p_0^{RP} = \frac{1}{N+M} \cdot \sum_{s=1}^{M+N} (p_e)^{m_s} \tag{5}$$

Depending on the values of r, M and N, several cases can be distinguished. We present results for the two most current cases:

Case 1: $r < M, N$. This means that the number of retransmissions does not always cover the δ_g period (which counts N states). In this case, p_0^{RP} can be written a follows:

$$\begin{aligned} p_0^{RP} &= P\left[s \in N \wedge (s+r) \in N\right] + P\left[s \in M \wedge (s+r) \in M\right] \\ &\quad + P\left[s \in M \wedge (s+r) \in N\right) + P\left[s \in N \wedge (s+r) \in M\right] \\ &= \frac{N-r+1}{M+N} + \frac{M-r+1}{M+N} \cdot {p_e}^r + \frac{1}{M+N} \sum_{i=1}^{r-1} {p_e}^i + \frac{1}{M+N} \sum_{i=1}^{r-1} {p_e}^{r-i} \\ &= \frac{N-r+1}{M+N} + \frac{M-r+1}{M+N} \cdot {p_e}^r + \frac{2}{M+N} \cdot \frac{p_e - {p_e}^r}{1-p_e} \end{aligned} \tag{6}$$

Case 2: $N < r < M$. . This is the most common case, where the number of repetitions is chosen to cover the entire inter-frame period. However, a reception is still not certain in this case, because of the decoding error p_e. In this case:

$$\begin{aligned} p_0^{RP} &= P\left[s \in M \wedge (s+r) \in M)\right] + P\left[s \in M \wedge (s+r) \in N\right] \\ &\quad + P\left[s \in N \wedge (s+r) \in M\right] \\ &= \frac{M-r+1}{M+N} \cdot {p_e}^r + \frac{1}{M+N} \sum_{i=1}^{r-1} {p_e}^i + \frac{1}{M+N} \sum_{i=1}^{N} {p_e}^{r-i} \\ &= \frac{M-r+1}{M+N} \cdot {p_e}^r + \frac{1}{M+N} \cdot \frac{p_e - {p_e}^r}{1-p_e} + \frac{1}{M+N} \sum_{i=1}^{N} {p_e}^{r-i} \\ &= \frac{M-r+1}{M+N} \cdot {p_e}^r + \frac{1}{M+N} \cdot \frac{p_e - {p_e}^r}{1-p_e} + \frac{1}{M+N} \cdot \frac{{p_e}^{r-N}(p_e - {p_e}^N)}{1-p_e} \end{aligned} \tag{7}$$

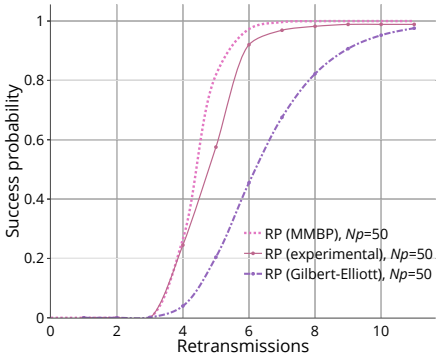

Figure 3: Probability to successfully receive N_p packets for the RP strategy. Dotted-lines show analytical results for the Gilbert-Elliott and MMBP models, while plain lines represent experimental results.

We compare the analytical results given by the two aforesaid models to experimentation results (obtained using our testbed described in [19]) in Fig. 3. This figure shows the success probability of receiving a message of $N_p = 50$ packets of data, as a function of the number of retransmissions r.

The very different results between the Gilbert-Elliott model and the experimentation confirms that the stochastic transition assumptions of this model are quite far from reality. On the other hand, MMBP approximates quite well the experimental behavior, highlighting the need for this more complex, but finer grained model.

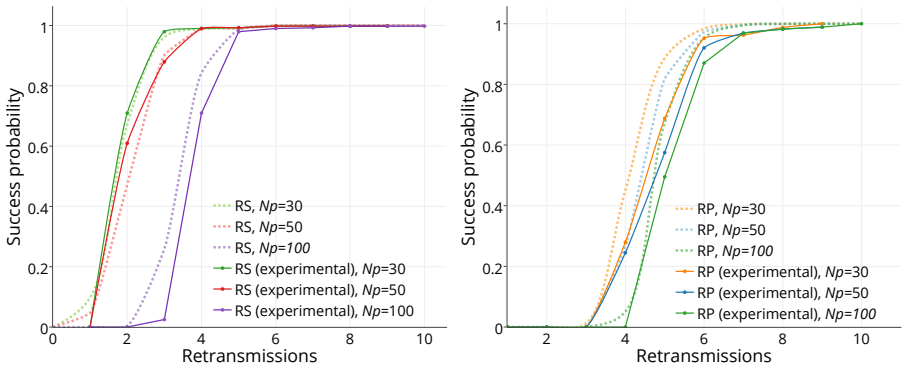

Figure 4: Probability to successfully receive N_p packets for the RS (left) and RP (right) strategies as a function of the number of retransmissions r. Dotted-lines show analytical results while plain lines represent experimental results.

3.3 Repeat Sequence

A different approach to improve reliability is the Repeat Sequence (RS) strategy, consisting in the transmission of a sequence of N_p packets, repeated r times. In contrast with the previous mechanism, RS does not try to cover the inter frame time at the packet level, and it does not ensure that a packet is received before sending the next one. Instead, the reliability and presumed efficiency is based on the fact that the probability of losing the same packets over different transmitted sequences is low.

In the case of an RS strategy with a sequence of N_p packets retransmitted r times, the probability of receiving a packet at least once can be written as $p_s^{RS} = 1 - p_0^{RS}$. Using the MMBP model, the probability of failing to receive a packet r times in a row, p_0^{RS}, can be written as:

$$p_0^{RS} = \frac{1}{M+N} \cdot \sum_{s=1}^{M+N} \prod_{j=0}^{r-1} p_0(s + rN_p) \tag{8}$$

3.4 Evaluation results

We use our MMBP analytical model to study the RP and RS strategies by focusing on the probability of delivering the entire quantity of information in a given number of transmissions. We provide both analytical and experimental results, allowing us to validate the proposed MMBP model.

Fig. 4 shows, for the two mechanisms, the probability of integrally receiving N_p packets of data as a function of the number of retransmissions r. In this figure, we set $M = 5$ and $N = 2$; these values are in line with the packet length, the transmitter frequency and the camera capture interval experimentally observed for a distance of 5 cm between LED and camera. The results show quite a nice fit between the analytical and experimentation results, despite the assumptions required by our MMBP model.

To better understand the performance of the two retransmission strategies, we compare them in Fig. 5. This figure shows that, for the RS strategy, 3 retransmissions are needed to achieve a reception probability higher than 0.9, while this value raises to 6 for the RP strategy. On the right side of the figure, we show that the performance of the two strategies depends on the ratio between the number of S_{ON} and S_{OFF} states, $M : N$. When this ratio changes from $5 : 2$ to $2 : 5$, which practically corresponds to increasing the distance between the LED and the camera, RP gives better results than RS. Indeed, for the RS method, the success probability sharply decrease when $M < 3$ and stays below 0.6 even for 10 retransmission.

Practically, this means that RP is more suitable when the distance between the LED and the camera is higher, while RS is better for short communication distances. This phenomenon was previously unknown in the research community, but it is straightforward to study with our analytical model

4 ROI MODEL

An important phenomenon in LED-to-camera communications comes as a direct consequence of the distance between the LED and the camera. Indeed, as this distance increases, the size of the region of interest (ROI) in the picture reduces and, as a consequence, cuts down the number of messages that the camera can receive per frame, i.e. the M states in Fig. 1. To include this performance factor into our model, we propose an analytical function that gives the ratio between the ROI and the picture size. In the model discussed in Subsec. 3.1, this is the ratio of M states in the $M + N$ states.

We apply photogrammetry rules to give the ROI ratio as a function of the distance d, the LED size l, the camera CMOS sensor size ss, the image size on the sensor i and the camera focal distance fc. According to the optical system depicted in Fig 6, basic lens optic rules give the following Eq. 9:

$$\frac{i}{fc} = \frac{l}{d} \iff i = \frac{l \cdot fc}{d} \tag{9}$$

To obtain the ROI as the ratio of the total number of pixels in the picture, we need as input the CMOS sensor size ss. We apply the min function to normalise the $ROI \in [0, 1]$ even if the image size on the sensor, i, is larger than the sensor size, ss. The result is given in the following Eq. (10):

$$ROI = \min\left(1, \frac{l \cdot fc}{d \cdot ss}\right) \tag{10}$$

To validate the results given by Eq. 10, we measure the ROI experimentally, using the testbed described in [19], for distances from 0 to 40 cm. Fig. 7 plots in orange the ROI ratio we observed

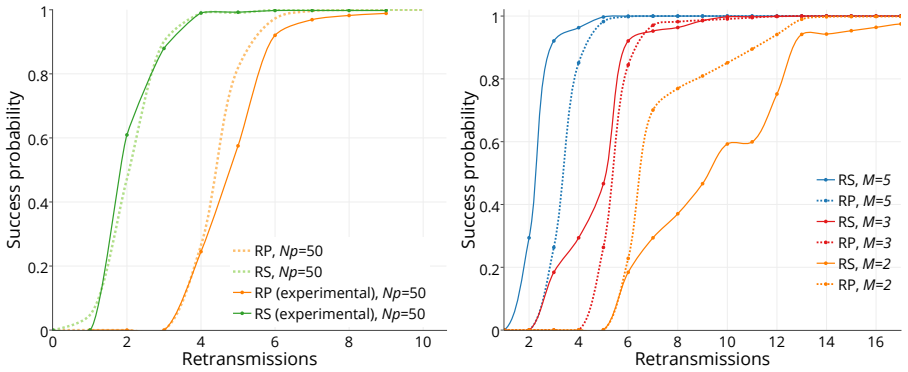

Figure 5: Comparison between RS and RP. On the left, analytical and experimental results for $M = 5$ and $N = 2$. On the right, analytical results when $M + N = 7$, but the $M : N$ ratio changes. In both cases, $N_p = 50$.

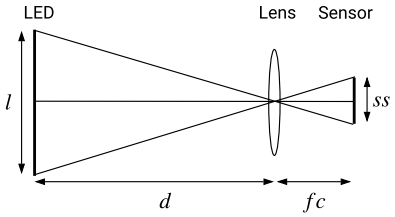

Figure 6: Formation of an image on a sensor by a converging lens.

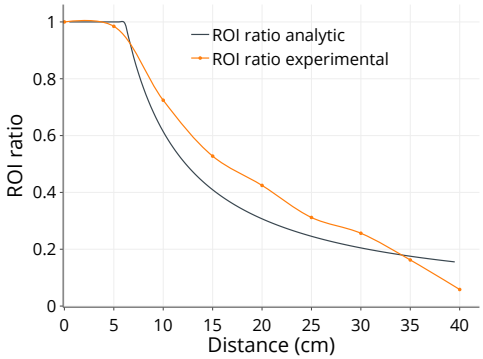

Figure 7: ROI as a function of distance. The orange line shows experimental results, while the green line represents analytical results given by Eq. 10.

during our experiments and in green the analytical results computed with a Nexus 5 sensor with the following characteristics: $fc = 35$, $ss = 5.7$ and $l = 10$. This shows that the analytical curve approximates quite well the experimental ROI ratio. However, we notice that the experimental results are better for a distance between 10 and 30 cm, and they become worse than the model at 35 cm. In fact, the light radiance on the camera lens, that our model does not take into account, artificially increases the LED size on the picture when the camera is close to the LED. The difference at larger distance is a consequence of the ambient light which was measured at 650 lux during the experiments, also neglected in Eq. (10).

The ROI model described in this section and the MMBP reception model validated in the previous section are the basis of the simulator implementation discussed in the following.

5 THE CAMCOMSIM SIMULATOR

As discusse in Sec. 2, the simulation of LED-to-camera communication remains completely unexplored in the field. Our work is the first such effort reported in the literature, making *CamComSim* the first implementation of a LED-to-camera VLC simulator.

5.1 Simulator Implementation

5.1.1 Software architecture. *CamComSim* is an event-driven LED-to-camera simulator developed in Java, which makes it easy to maintain and distribute code, and it provides built-in multi-platform compatibility for systems with a Java Virtual Machine. Fig. 8 shows the *CamComSim* software architecture that consists of a simulator kernel class and four core packages. For interested readers, *CamComSim* is already available as an open-source software under Apache license at http://vlc.project.citi-lab.fr/camcomsim.

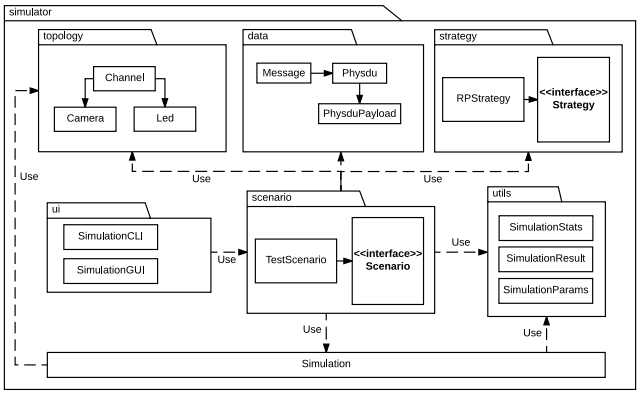

Figure 8: The *CamComSim* software architecture and packages dependency graph.

Param.	Description	Default Value
d	Distance between camera and LED (cm)	5
l	LED size (mm)	4
d_g	Camera inter-frame gap ratio	0.1
p_e	Decoder PHY-SDU Error Rate	0.001
f	Modulation frequency (Hz)	8000
P	`PhysduPayload` Header length (bit)	8
H	`PhysduPayload` Payload length (bit)	16
r	PHY-SDU repeat number	1
G	Message size (bytes)	50

Table 1: Simulator parameters and default values.

The `topology` package groups classes that describe the system components: `Led`, `Camera` and `Channel`. The classes in the `data` package implement the data encapsulation. For this, a `Message` is a set of PHY-SDU that encapsulates a `PhysduPayload`. A `Packet` is a `PhysduPayload` child class, with a sequence number as header and a payload that contains data. Before each simulation, a `Message` is created according to the user settings. The resulting set of `Physdu` is initialized with a `Packet` filled with arbitrary data in the payload (real data could be used if available) and a unique sequence number in the header.

The broadcast strategy abstraction is given in the `strategy` package. Here, the `Strategy` interface lets the users implement their transmission strategy. This package also contains the straightforward `ReapeatPhysduStrategy` (RP) implementation that consists in repeating each PHY-SDU r times, one after the other, as described in Sec. 3.2. When the last PHY-SDU of the message is reached, the process is repeated from the beginning.

Finally, the `scenario` package proposes an interface to build a simulation, wiring together the `Message`, the `Channel`, the `Led`, the `Camera` and the `Strategy` with the `Simulator` kernel. Besides, the package `utility` provides helper classes used to compute the simulation results statistics, format and save the results as a JSON file and load or save the simulation parameters. The `ui` package contains a command-line interface (CLI) used to run a simulation scenario.

5.1.2 Simulator parameters. Our simulator exposes a set of finely grained parameters to describe the LED-to-camera communication system behavior. Table 1 shows the parameters we use in this work and expose through the *CamComSim* CLI. The performance of the LED-to-camera communication is significantly affected by the distance d, the IFG noted δ_g in Fig. 2, and the LED size l. The values of these parameters should therefore be carefully chosen, according to the available hardware and envisioned scenario.

Further parameters, introduced in Sec. 4, that refer to the CMOS sensor characteristics, are optional but can be considered to refine the channel model, as they impact the ROI. However, smartphone manufacturers rarely provide this information, e.g. regarding the sensor size ss and the focal distance fc.

The PER p_e is the consequence of the errors occurring in a M state when a PHY-SDU is well included in a picture but is wrongly decoded by the smartphone. These errors are bits substitutions induced by interference, low SNR, and artifacts on the picture. This value varies from a smartphone to another, but we did not observe major differences in our tests.

The `PhysduPayload` payload size P and the `PhysduPayload` header size H configure the data encapsulation mechanism. Given these two settings, the PHY-SDU size is computed as $P+H+SYNC+PB$, where $SYNC$ is the PHY-SDU delimiter symbol (4 bits in our tests), and PB is the number of parity bits (2 bits in our settings). This PHY-SDU size, along with d, l, δ_f and the modulation frequency f, determines the number of the M time slots in Fig. 2.

A transmission strategy among the `Strategy` interface implementations needs to be chosen as well. For now, we have implemented and considered only the RP strategy, for which the parameters are the size of the message to broadcast, G, and the number of consecutive PHY-SDU emissions, r.

5.1.3 Kernel Implementation. The *CamComSim* kernel is implemented in the `Simulator` class. Its role is to produce PHY-SDU emission events (TX) and manage their result. The number of events, i.e. the number of PHY-SDU sent, is noted c and is updated at runtime. At each clock tick, c is incremented, a TX event is created and processed as follow: (1) the next PHY-SDU in the transmission strategy queue is associated with this event; (2) considering p_e, f, P, H, c, the channel response function gives the event result. The result is one among: *reception success*, *reception with errors* or *loss during IFG*; (3) this result is stored in a list to further determine if all the PHY-SDU forming a message are received.

The simulator loops over (1), (2) and (3) until the stop condition is met, i.e. c has reached the maximum number of PHY-SDU emissions or the complete message is received. The simulation is repeated n_r times using the Java class for multi-threading purpose `ThreadPoolExecutor`. Finally, the simulations results and statistics are saved in a JSON file for further processing.

5.2 CamComSim validation

In this section, we present LED-to-camera simulation results given by *CamComSim*. To assess the correctness of our simulator, we conduct a series of experiments with the testbed presented in [19]. We set the emitter symbol rate to 8 kHz and place it in standard indoor illumination conditions, near a window and illuminated with neon lights. The illuminance has been measured with a luxmeter at around 650 lux. We compare the testbed performance with the results given by *CamComSim* for a set of key parameters: the message size G, the number of consecutive PHY-SDU emissions r, the distance d and the PHY-SDU payload length P.

5.2.1 PHY-SDU Retransmission. As discussed in Sec. 3.2, to face the IFG bits erasure and ensure that all the packets are well received, a possibility is to transmit consecutively each PHY-SDU r times in a row. The `ReapeatPhysduStrategy` class in *CamComSim* implements this retransmission strategy.

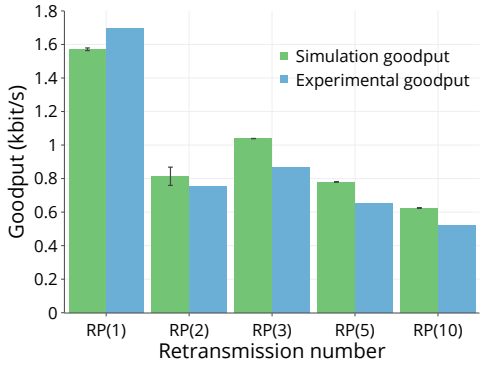

Figure 9: The experimental goodput (blue) compared with the simulation goodput (green) as a function of the number of consecutive PHY-SDU emissions. The bars on top are 95% confidence intervals.

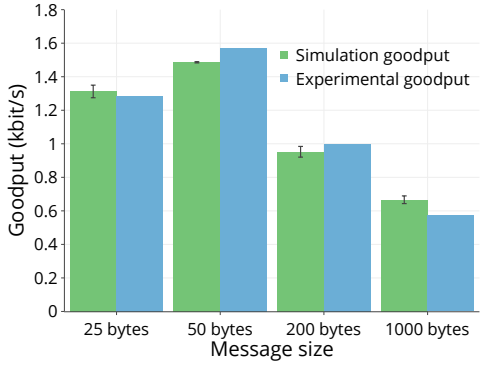

Figure 10: The experimental goodput (blue) compared with the simulation goodput (green) for different message size (G, bytes).

Fig. 9 shows the goodput at 5 cm for different values of PHY-SDU consecutive retransmissions r, with $G = 50$, $P = 19$, $H = 5$. When each PHY-SDU has been transmitted r times, the message transmission restarts, until the message is completely received. To avoid infinite loops, we stop the simulation when 50000 PHY-SDU are sent, even if the message is not received entirely. In such case, we consider the goodput is 0.

The results highlight that the simulation and testbed goodput follow the same tendency when r varies. The best case is when $r = 1$, for which the goodput is 1.6 kbit/s according to *CamComSim* and 1.7 kbit/s for the testbed, an estimation error of only 6%. Based on these results, for all the simulations that follow we use the RP strategy implementation with $r = 1$.

5.2.2 Message size. We now consider the impact of the message size G on the goodput at a 5 cm distance, with $r = 1$ and a 24 bits length PHY-SDU. Fig. 10 shows that *CamComSim* results are very close to those of the testbed, confirming that the simulator well considers the impact of the message size. The goodput reduces when the message size increases, as the RP strategy leads to a large

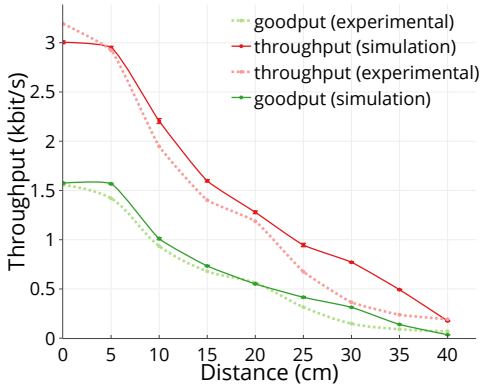

Figure 11: The throughput (red) as a function of the distance, compared with the goodput (green). Dotted-lines show experimental results while plain lines represent simulation results.

number of useless transmissions: the simulator gives 1.6 kbit/s of goodput for $G = 50$ bytes, while this falls to 670 bit/s when $G = 1000$ bytes. These results differ from the testbed in no more than 7%.

5.2.3 Distance. Fig. 11 shows the goodput and the throughput as a function of the distance, when the LED broadcasts a 50 bytes message. The PHY-SDU payload is set to 24 bits, with $P = 19$ and $H = 5$. The results show a good match between the simulation and real life results. At 10 cm, *CamComSim* gives 2.2 kbit/s of throughput, against 1.94 kbit/s experimentally. The results are closer for the goodput: 0.94 and 1.0 kbit/s respectively for simulation and experimentation, that is only 6% of difference.

This section highlights that *CamComSim* gives results very close to the testbed for all the parameters we have studied. The difference is around 10% and often less. For all the cases we consider, *CamComSim* respects the behavior of the LED-to-camera communication system implemented by the testbed.

5.3 Use case

In this section, we detail a case study for *CamComSim*, applied to a real life scenario. A common issue with cheap consumer electronics is the lack of diagnostics when a dysfunction happens. Manufacturers often blink the state LED with a pattern and color that match with an error code. Such a mechanism is easy to implement but leads to inaccurate diagnostics. For these cases, we propose to benefit from this LED to perform LED-to-camera communication and broadcast a log file that would include helpful information to diagnose a dysfunction. We consider a worst case file size of 1 kbyte that is large enough for events history or debug traces.

Fig. 12 compares the goodput given by *CamComSim* with the goodput that our testbed achieved for the transfer of a 1 kbyte log file as a function of the number of PHY-SDU retransmissions r. Note that this is equivalent to $G = 1000$ bytes in Fig. 10. The transmission restarts until the message is received. The left side plot shows the results when the LED and the smartphone are 5 cm apart, while the distance is 10 cm on the right side figure. At 5 cm, the simulation brings out that, to obtain the higher goodput, the emitter should

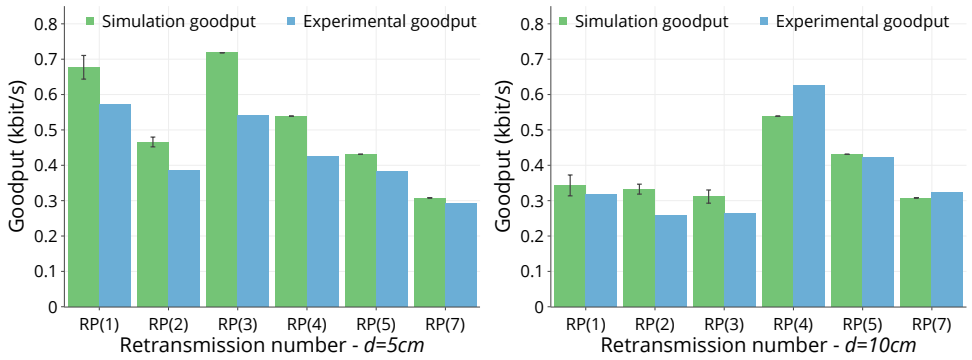

Figure 12: The experimental goodput (blue) compared with the simulation goodput (green) for the use case as a function of the number of consecutive PHY-SDU emission at 5 cm (left) and 10 cm (right).

send each PHY-SDU one or three times consecutively, i.e. $r = 1$ or $r = 3$. The goodput is respectively 680 and 720 bit/s in these cases. This finding is similar to the testbed, where the goodput is 570 bit/s when $r = 1$ and 540 bit/s when $r = 3$.

Because the ROI decreases with the distance, the behavior is different when the smartphone is 10 cm far from the LED. In this situation, $r = 4$ stands out clearly to be the best choice both for the simulation and the experiments. The goodput then becomes 620 bit/s on the testbed and 540 bit/s with *CamComSim*.

Since the results are very close to the reality, using *CamComSim* highly reduces the search space for the experimental optimization of a system. As shown by these results, the best value for r can be decided using simulations only, removing the need for a lengthy experimental campaign.

6 CONCLUSION

In this paper, we introduced *CamComSim*, the first simulator for the design, the prototyping and the development of protocols and applications for LED-to-camera communication. Our event driven simulator is based on an MMBP channel model, and it relies on a standalone Java application that is easily extensible through a set of interfaces. We have validated *CamComSim* comparing simulation results with the performance reached by a real life testbed. Then, we illustrated with a practical use case the complete usage of *CamComSim* to tune a broadcast protocol that implements the transmission of a 1 kbyte log file. The results highlight that our simulator is very precise and can predict the performance of a LED-to-camera system with less than 10% of error in most cases. The availability of accurate performance evaluation tools offers a great ease of use and the opportunity to tune protocols without the burden of always realizing experiments on a testbed.

REFERENCES

[1] C. Danakis, M. Afgani, G. Povey, I. Underwood, H. Haas. "Using a CMOS Camera Sensor for Visible Light Communication". *Proc. IEEE OWC 2012*, Anaheim, CA, USA, Dec. 2012.

[2] T. Nguyen, Y.M. Jang. "High-speed Asynchronous Optical Camera Communication using LED and Rolling Shutter Camera". *Proc. ICUFN 2015*, Sapporo, Japan, Jul. 2015.

[3] P.H. Pathak, X. Feng, P. Hu, P. Mohapatra. "Visible Light Communication, Networking, and Sensing: A Survey, Potential and Challenges". *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2047–2077, Oct. 2015.

[4] H.-Y. Lee, H.-M. Lin, Y.-L. Wei, H.-I. Wu, H.-M. Tsai, C.-J. Lin. "RollingLight: Enabling Line-of-Sight Light-to-Camera Communications". *Proc. ACM MobiSys 2015*, New York, NY, USA, May 2015.

[5] J. Hao, Y. Yang, J. Luo. "CeilingCast: Energy Efficient and Location-bound Broadcast through LED-camera Communication". *Proc. IEEE INFOCOM 2016*, San Francisco, CA, USA, Apr. 2016.

[6] H. Du, J. Han, X. Jian, T. Jung, C. Bo, Y. Wang, X.-Y. Li. "Martian: Message Broadcast via LED Lights to Heterogeneous Smartphones". *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1154âĂŞ1162, May 2017.

[7] J. Ferrandiz-Lahuerta, D. Camps-Mur, J. Paradells-Aspas. "A Reliable Asynchronous Protocol for VLC Communications Based on the Rolling Shutter Effect". *Proc. IEEE GlobeCom 2015*, San Diego, CA, USA, Dec. 2015.

[8] C. Zhang, X. Zhang. "LiTell: Robust Indoor Localization using Unmodified Light Fixtures". *Proc. ACM MobiCom 2016*, New York, NY, USA, Oct. 2016.

[9] S. Zhu, X. Zhang. "Enabling High-Precision Visible Light Localization in TodayâĂŹs Buildings". *Proc. ACM MobiSys 2017*, Niagara Falls, NY, USA, Jun. 2017.

[10] T. Li, C. An, Z. Tian, A.T. Campbell, X. Zhou. "Human Sensing Using Visible Light Communication". *Proc. ACM MobiCom 2015*, Paris, France, Sep. 2015.

[11] S. Zhu, C. Zhang, X. Zhang. "Automating Visual Privacy Protection Using a Smart LED". *Proc. ACM MobiCom 2017*, Snowbird, UT, USA, Oct. 2017.

[12] M. Sharif, A. Sadeghi-Niaraki."Ubiquitous Sensor Network Simulation and Emulation Environments: A Survey". *Journal of Network and Computer Applications*, vol. 93, pp. 150âĂŞ181, Sep. 2017.

[13] D. Tagliaferri, C. Capsoni. "Development and Testing of an Indoor VLC Simulator". *Proc. IEEE IWOW 2015*, Istanbul, Turkey, Sep. 2015.

[14] A. Aldalbahi, M. Rahaim, A. Khreishah, M. Ayyash, R. Ackerman, J. Basuino, W. Berreta, T.D. Little. "Extending ns3 to Simulate Visible Light Communication at Network-level". *Proc. IEEE ICT 2016*, Thessaloniki, Greece, May 2016.

[15] C. Ley-Bosch, R. Medina-Sosa, I. Alonso-Gonzalez, D. Sanchez-Rodriguez. "Implementing an IEEE802.15.7 Physical Layer Simulation Model with OMNET++". *Proc. DCAI 2015*, Salamanca, Spain, Jun. 2015.

[16] A. Musa, M. D. Baba, H. M. Haji Mansor. "The Design and Implementation of IEEE 802.15.7 Module with ns-2 Simulator". *Proc. IEEE I4CT 2014*, Lisbon, Portugal, May 2014.

[17] S. Ozekici. "Markov Modulated Bernoulli Process". *Mathematical Methods of Operations Research*, vol. 45, no. 3, pp. 311âĂŞ-324, Mar. 1997.

[18] E.O. Elliott. "Estimates of Error Rates for Codes on Burst-Noise Channels". *Bell System Technical Journal*, vol. 42, no. 5, pp. 1977âĂŞ-1997, May 1963.

[19] A. Duque, R. Stanica, H. Rivano, A. Desportes. "Unleashing the Power of LED-to-Camera Communications for IoT Devices". *Proc. ACM VLCS 2016*, New York, NY, USA, Oct. 2016.